determine if any of the components in the software package are shared with other software packages registered in the code store data structure (step **335**). If so, the package manager performs one of three basic actions depending on the version information stored in the code store data base entries for the component (step **337**) If the newly stored component is a newer version of the previously stored component (step **339**) and the code store data structure entry for the previously installed software package that references the older version does not indicate it is version dependent (step **341**), the package manager removes the older version from the directory in which it is stored (step **343**) and updates the code store data structure entry for the previously installed software package to point to the newer version (step **345**). If there is a version dependency noted, the package manager leaves the older version in the directory (step **347**).

[**0062**] If the newly stored component is older that the previously stored component (step **349**) and the software package does not indicate a version dependency (step **351**), the package manager removes the older version from the newly created directory (step **353**) and updates the code store data structure entry for the newly installed software package to point to the newer version (step **355**). As before, if there is a version dependency noted, the package manager does nothing to the older version (step **347**).

[**0063**] If the components are the same version, the package manager chooses one to remove from its directory (step **359**) and updates the corresponding entry to point to the other component (step **361**).

[**0064**] In the embodiment in which the components are stored in an uncompressed zip file, or the like, the package manager uses the file directory to find the component to be deleted within the file in the appropriate directory. The package manager can, alternately, actually delete the component from the file and update the file directory or mark the component entry in the file directory as deleted depending on the particular file structure employed to hold the components.

[**0065**] In an alternate embodiment, the package manager does not attempt to determine if there are mismatched versions installed and each software package uses the version of the component that is indicated by its entry in the code store data structure.

[**0066**] Execution

[**0067**] Referring next to **FIG. 3C, a** flowchart of a method to be performed by a client computer according to an exemplary embodiment of the invention when a software package registered through the package manager is executed in the runtime environment is shown. This method is inclusive of the steps or acts required to be taken by the software package manager.

[**0068**] When the user requests execution of the software package, the runtime environment invokes the package manager (step **370**) to locate the components necessary to run the software. The package manager matches the software package name to the corresponding entry in the code store data structure (step **371**) to determine the directory, or directories, holding the components (step **373**). In the embodiment in which the components are stored in an uncompressed zip file, or the like, the package manager uses the directory to find the particular components within the file. The package manager returns the location of the components to the runtime environment (step **375**).

[**0069**] Uninstall

[**0070**] Finally, referring to **FIG. 3D, a** flowchart of a method to be performed by a client computer according to an exemplary embodiment of the invention when a software package registered through the package manager is uninstalled is shown. This method is inclusive of the steps or acts required to be taken by the software package manager.

[**0071**] When the user wants to uninstall a software package from the local computer, a standard uninstall routine provided by the runtime environment invokes the package manager to update the code store data structure accordingly (step **380**). The package manager removes the corresponding software package entry in the code store data structure (step **381**). The package manager does not delete a component from the directory unless no other installed software package references it (step **383**).

[**0072**] In one embodiment, the package manager scans every entry in the code store data structure to determine if another the entry for another software package references the local directory holding the component in question. In a first alternate embodiment, the package manager creates and maintains a tree structure of all shared components, so it can quickly determine if the component is shared with another software package. In a second alternate embodiment, the component is not deleted at the time the software package is uninstalled but instead a "garbage collection" routine is periodically run by the package manager. The garbage collection routine uses the code store data structure to determine if a component is referenced by any of the software packages installed on the local computer. Those components which are not referenced are then deleted.

[**0073**] Code Store Data Structure

[**0074**] **FIG. 4** illustrates an exemplary embodiment of an entry in the code store data structure **400** suitable for use by the methods of the exemplary embodiments of the package manager described above. Each entry contains, at a minimum, five fields: a name field **401** for the distribution unit, a version field **403** for the distribution unit, a component field **405** that contains a list of the components in the distribution unit, a location field **407** that contains the location of the components on the client computer, and a source field **409** that contains a pointer to the source of the distribution unit, such as a URL for a downloaded distribution unit. If a component is a platform-specific ("native code") file, such as a Microsoft Windows DLL, the file name is the component is stored in the component field. If a component is a package of Java classes, the component field contains the name of the Java package. In the case of a Java package, the code store entry has the following additional fields: a component version field **411** (which may be different from the version of the distribution unit; both are used by the package manager in resolving version dependencies), and a component type field **413** that indicates what type of Java classes the package contains, i.e., system, application, etc., both shown in phantom in **FIG. 4** An optional data signature field **415**, also shown in phantom, contains a digital signature affixed to the distribution unit, if it was signed. As well be familiar to one of skill in the art, the